

Multithreading

Introduction



Terminology

- **Process**

- An instance of a program currently executing
- Assigned its own resources and memory space
- Contains at least one **thread of execution**

- **Thread**

- Exists within a process and shares its resources
- Similar to a lightweight process

<http://docs.oracle.com/javase/tutorial/essential/concurrency/procthread.html>



Terminology

- **Concurrency**

- Performing more than one action simultaneously
- May be applied to processes or threads

- **Multithreading**

- Running multiple threads per process
- Create **worker threads** to handle specific tasks

<http://docs.oracle.com/javase/tutorial/essential/concurrency/index.html>



Multithreading

- Start with a **large** and **parallelizable** problem
 - i.e. can break a large problem into smaller tasks that can be completed simultaneously
- Create **worker threads** to handle smaller tasks
- Use **synchronization** to get final results from workers



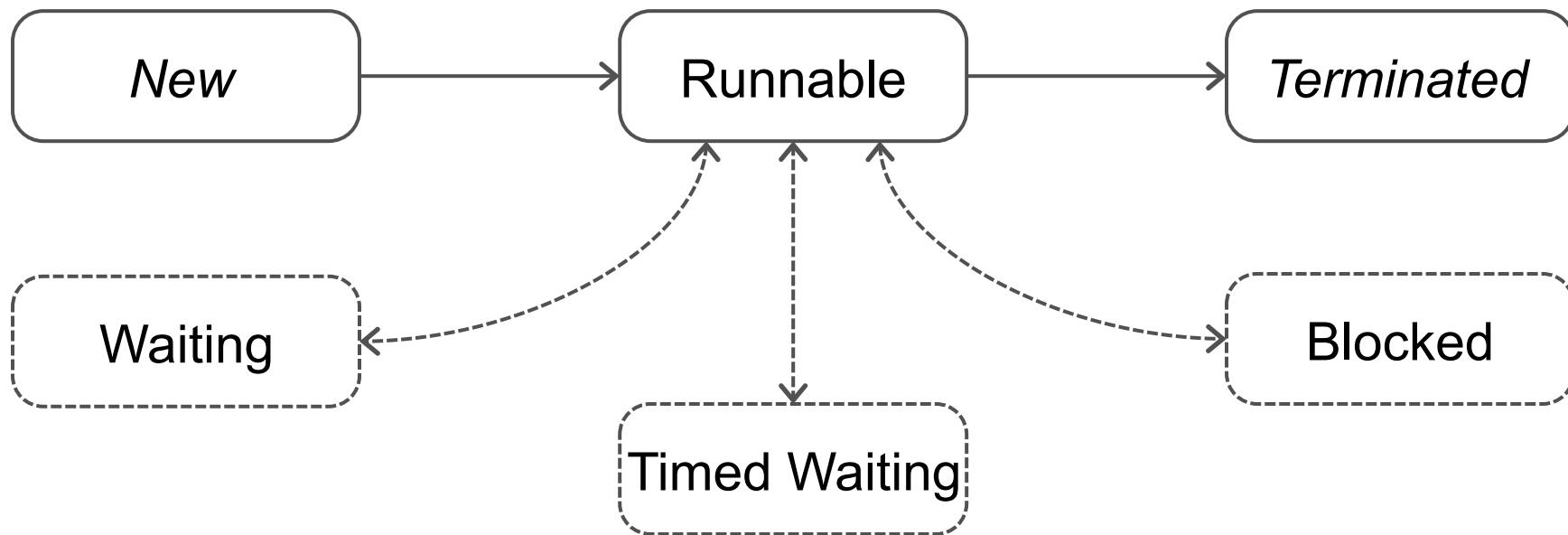
Thread Lifecycle

- Create a **new** thread and initialize members
 - Once complete, thread becomes runnable
- A **runnable** thread is ready to perform work
 - Might be **waiting** for something, or be **blocked** from a resource that is busy
- When work is complete, thread is **terminated**
 - Data members still around in memory

<http://www.ibm.com/developerworks/java/tutorials/j-threads/section3.html>



Thread States



<http://www.ibm.com/developerworks/java/tutorials/j-threads/section3.html>

Multithreading Classes

- Object Class
 - notify(), notifyAll(), wait()
- Runnable Interface
 - run()
- Thread Class
 - start(), join(), sleep(), and others

<http://docs.oracle.com/javase/8/docs/api/java/lang/Thread.html>



Multithreading in Java

- **Creating Threads**

- Extend the Thread class and override run()
- Implement the Runnable interface and pass to Thread constructor

- **Managing Threads**

- Manually (call start(), join(), etc. in code)
- Via a task executor (discussed later)

<http://docs.oracle.com/javase/tutorial/essential/concurrency/threads.html>



Obstacles

- Creating threads requires **time** and **resources**
 - For small amounts of work, may *slow* code
 - For large amounts of work, may *speedup* code
- Must **synchronize** access to **shared data**
- Order of operations is **non-deterministic**
 - Difficult to debug and replicate problems





CHANGE THE WORLD FROM HERE